Exam : **AB-400**

Title : Extending Microsoft Power Platform Solutions with Code and AI

Version : DEMO

1.An integration submits a top-level Create request for a Shipment row in a standard Dataverse table. Custom code must also update a related Booking row. Both Dataverse changes must commit or roll back together. A carrier service must be notified only after those changes commit.
Notification can complete later, and carrier failures must not reject shipment creation.
Which implementation should you choose?
A. Use a synchronous PreOperation plug-in to update Booking and a synchronous PostOperation plug-in to notify the carrier.
B. Use a synchronous PreValidation plug-in to update Booking and an asynchronous PostOperation plug-in to notify the carrier.
C. Use an asynchronous PostOperation plug-in to update Booking and then notify the carrier.
D. Use a synchronous PreOperation plug-in to update Booking and an asynchronous PostOperation plug-in to notify the carrier.
Answer: D
Explanation:
A synchronous PreOperation plug-in runs within the Shipment Create transaction, allowing a propagated exception to roll back both Dataverse changes. An asynchronous PostOperation plug-in runs after the record operation completes and outside that transaction, placing carrier notification after commit. Synchronous PostOperation still participates in the transaction: it follows the main operation, not the transaction's commit.
A - Incorrect: The Booking update has the required transaction boundary, but synchronous PostOperation still runs inside the Create transaction. The carrier call would occur before commit, violating the required notification timing.
B - Incorrect: For this top-level request, PreValidation precedes the main transaction, so its Booking update does not share the required rollback boundary. A nested operation invoked by a transactional extension can already be in a transaction at PreValidation, but that exception does not apply here.
C - Incorrect: Asynchronous PostOperation runs after Shipment creation completes. A subsequent failure updating Booking cannot roll back the original Shipment creation, so the two Dataverse changes are not atomic.
D - Correct: Synchronous PreOperation keeps the Booking update within the original Create transaction. Asynchronous PostOperation places carrier notification outside that transaction and after successful completion, satisfying both boundaries.

2.A standard Dataverse Order table has a decimal Amount column. Both the model-driven app and a Web API integration must receive an error before an Order with a negative Amount is committed.
An active business rule has this configuration:
Scope: All Forms
Condition: Amount < 0
Action: Show an error message on Amount
The model-driven form blocks a save with Amount = -10. An ordinary Dataverse Web API Create request with Amount = -10 succeeds.
The architect considers the existing validation sufficient for both callers.
Which revision corrects the enforcement gap?
A. Change the business rule's scope to the main Order form, retaining its condition and error action.
B. Change the business rule's scope to Entity, retaining its condition and error action.

C. Keep the business rule and add an OnSave JavaScript check for negative amounts to all Order forms.
D. Keep the business rule and add an asynchronous PostOperation plug-in on Order Create to check for negative amounts.

Answer: B

Explanation:

The same negative amount is blocked through the form but accepted through the Web API. The active rule's condition works, but its scope does not cover both callers. All Forms scope applies to model-driven forms; Entity scope also applies on the server. Changing the scope therefore provides server enforcement while retaining the supported numeric condition and error action. An asynchronous server check addresses a different boundary: it runs after the record operation completes and cannot reject the original Create request.

A - Incorrect: A specific-form scope applies only to that model-driven form. Assigning the rule to the main Order form narrows its form coverage and does not enforce it for Web API requests.

B - Correct: Entity scope applies to model-driven forms and the server. The numeric condition and error action can remain declarative, so changing the scope supplies the missing enforcement boundary without requiring custom code.

C - Incorrect: An OnSave script provides client-side validation when the relevant form event runs. Direct Web API requests do not execute that form event, so the integration remains unprotected.

D - Incorrect: An asynchronous PostOperation plug-in runs after the record operation completes, outside the original transaction. It can detect a negative amount afterward, but it cannot prevent that amount from being committed by the original Create request.

3.A Power Platform app calls an HTTP-triggered Azure Function to generate a report that normally takes six minutes. The app requires the initiating call to return within three seconds. Users must later retrieve either the report or failure details, including after the function host has restarted.
The six-minute processing duration is within the selected hosting plan's execution limit.
Which implementation should you choose?

A. Generate the report in the HTTP-triggered function, persist its outcome, and return a request ID with the report after processing ends.
B. Queue the request and return a request ID after enqueueing succeeds. Use a queue-triggered worker to persist status and outcomes in durable storage for a status endpoint.
C. Queue the request and return a request ID after enqueueing succeeds. Use a queue-triggered worker to keep status and outcomes in process memory for a status endpoint.
D. Start report processing as a background task in the HTTP-triggered function and return a request ID immediately. Have that task persist status and outcomes in durable storage for a status endpoint.

Answer: B

Explanation:

The report cannot finish within the initiating call's three-second deadline. Successful queue submission can be acknowledged with a request ID while a queue-triggered worker performs the work. A status endpoint backed by durable status and outcome data lets the app retrieve the report or failure details after host restarts. The initial acknowledgment means accepted, not completed. The worker must still respect its hosting plan's execution limit.

A - Incorrect: The hosting plan permits six minutes of processing, but waiting for the report before returning violates the app's three-second response deadline. Persisting the outcome does not shorten

the initiating call.
B - Correct: The HTTP invocation acknowledges accepted work without waiting for report generation. The separate worker performs tracked processing, and durable status and outcomes provide a retrieval path that survives host restarts.
C - Incorrect: The queue separates processing from the initiating call, but process memory does not preserve status or results across host restarts. Durable request submission does not make the worker's in-memory outcomes durable.
D - Incorrect: Azure Functions does not track background tasks after the function invocation returns. Host shutdown can interrupt report generation even when the task intends to save its outcome in durable storage. A separate queue-triggered invocation provides the required processing boundary.

4.You are designing a Dataverse invoice-processing solution with two components:
- Review app: Employees sign in and approve invoices. Each update must be authorized using the signed-in employee's Dataverse permissions.
- Import service: A scheduled process imports invoices overnight without a signed-in employee. It must operate under its own application authority without depending on an employee account.
Which identities should be the Dataverse callers for these components?
A. Use delegated employee identities as the Dataverse callers for both components.
B. Use an application user as the Review app's Dataverse caller and a delegated employee identity as the Import service's caller.
C. Use a delegated employee identity as the Review app's Dataverse caller and an application user as the Import service's caller.
D. Use application users as the Dataverse callers for both components.
Answer: C
Explanation:
The caller's relationship to the work determines the identity choice. The Review app performs updates under the signed-in employee's Dataverse permissions, so delegated user access fits. The Import service operates independently of employees, so its caller should be an application user associated with its registered application and assigned appropriate Dataverse security roles. Application-user access for both components would support the unattended import but would make the Review app's caller the application user rather than the employee. See [Use OAuth authentication with Microsoft Dataverse](https://learn.microsoft.com/en-us/power-apps/developer/data-platform/authenticate-oauth).
A - Incorrect: Delegated employee access fits the Review app. Using it for the Import service would make the unattended integration depend on an employee identity instead of its own application authority.
B - Incorrect: This reverses the required caller contexts. The Review app needs the employee's identity, while the Import service needs an identity independent of employees.
C - Correct: Delegated employee access retains the Review app's required user context. An application user provides the Import service with its own Dataverse identity for unattended operations.
D - Incorrect: Application-user access fits the Import service. Making an application user the Review app's caller would use that application's Dataverse permissions rather than the signed-in employee's permissions.

5.An unattended reporting integration uses Microsoft Entra client credentials to read the organization-owned Shipping Zone table through the production Dataverse Web API.

The request fails with a table-privilege error.
A developer confirms these findings:
- The access token is valid, unexpired, and issued for the production Dataverse environment.
- The application user is active in that environment, and its application ID matches the calling application.
- No security roles are granted to the application user directly or through teams.
Which change addresses the failure?
A. Replace the application's client secret and acquire another Dataverse access token.
B. Assign the production application user a Dataverse security role with Read privilege on the Shipping Zone table.
C. Grant admin consent for the app registration's delegated Dynamics CRM user_impersonation permission.
D. Assign the developer's user account a Dataverse security role with Read privilege on the Shipping Zone table.
Answer: B
Explanation:
The valid token and matching active application user establish the caller's identity in the intended environment. The failure concerns Dataverse authorization: the application user has no security role granting the required table Read privilege. Assigning an appropriate Dataverse role addresses that missing grant. Delegated API consent does not supply the application user's table privileges. Microsoft documents application-user role assignment separately from application credentials in [Use OAuth authentication with Microsoft Dataverse] (https://learn.microsoft.com/en-us/power-apps/developer/data-platform/authenticate-oauth).
A - Incorrect: Replacing credentials addresses a credential problem. The application already obtains a valid token, and obtaining another token does not add the missing Dataverse privilege.
B - Correct: The application user is the caller, and its effective role grants are empty. Assigning a Dataverse security role with Read privilege on Shipping Zone supplies the required authorization.
C - Incorrect: The delegated user_impersonation permission concerns access on behalf of a user. Granting its consent does not assign Dataverse table privileges to this application user.
D - Incorrect: The integration uses application credentials. A role assigned to the developer's user account does not become a role of the application user.